Implementing logistic Regression with Vectorization.

Vectorization:

for i = 1 to m:

$z^{(i)} = w^T x^{(i)} + b \longrightarrow Z = w^T X + b$

$a^{(i)} = \sigma(z^{(i)}) \longrightarrow A = \sigma(Z)$.

$dz^{(i)} = a^{(i)} - y^{(i)} \longrightarrow dZ = A - Y$

$dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)} \longrightarrow$

$dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)} \longrightarrow$ $dw = \frac{1}{m} X dz^T$

⋮

$db \mathrel{+}= dz^{(i)} \longrightarrow db = \frac{1}{m}$ np.sum(dz).

$w := w - \alpha\, dw$ update (learning steps).

$b := b - \alpha\, db$.

General Principle for python Broadcasting.

(m, n) matrix $+ - * /$ (1, n) $\rightsquigarrow$ (m, n)

(m, 1) $\rightsquigarrow$ (m, n).

any operations of a col/row vector on another matrix would result in the operation being broadcasted to col/row of the matrix. result matrix will be in the same shape.

eg. (m, 1) + R

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + 100 = \begin{bmatrix} 101 \\ 102 \\ 103 \end{bmatrix}$$

python / numpy vectors.

a = np.random.randn(5)

This is neither a row or a col vector.

a.shape = (5,) $\longrightarrow$ "rank 1 array" Don't use.

a = np.random.randn(5, 1) $\longrightarrow$ a.shape = (5, 1)

a = np.random.randn(1, 5) $\longrightarrow$ a.shape = (1, 5).